

Aufgabe 1

Ändern Sie die Sprache **WHILE** ab, indem Sie anstelle des atomaren Ausdrucks `read` Anweisungen der Form `read I` zulassen. Die Semantik dieser Anweisung lautet informell: Die Ausführung von `read I` bewirkt eine Zuweisung des nächsten Eingabewertes an die Variable `I` und eine Verkürzung der Eingabedatei um das erste Element.

Formalisieren Sie die Semantik von `read I` denotationell.

Tobi: Idee von Flo etwas weiter gedacht:

$$\mathcal{C}[\underline{\text{read } I}](s, e, a) = \begin{cases} (S[n/I], e', a), & \text{falls } \mathcal{T}[\underline{\text{read}}] s e = (n, e') \\ \text{Fehler}, & \text{sonst} \end{cases}$$

Soll heißen: der nächste gelesene Eingabewert (n) ersetzt (I) im Speicher. (e') ist ein um den ersten Wert kürzeres (e).

Aufgabe 2

Erweitern Sie die Sprache **WHILE** um Anweisungen der Form

`for I := T_1 to T_2 do C`

. Formalisieren Sie die Semantik dieser Anweisungen denotationell.

Idee von Flo:

Das soll doch am Ende ne ganz simple Zählvariable werden denk ich mal, also:

$$\mathcal{C}[\text{while } I \neq T_2 \text{ do } C](s, e, a) = \mathcal{C}[C; \text{while } I \neq T_2 \text{ do } C; I = I + 1](s, e', a)$$

Solange I noch nicht T_2 entspricht, wird das Programm weiter ausgeführt.

Idee von Tobi:

Ja, die Idee ist gut, nur bissl anders aufgeschrieben:

$$\mathcal{C}[\text{for } I := T_1 \text{ to } T_2 \text{ do } C](s, e, a) = \begin{cases} \mathcal{C}[I = I + 1; C; \text{for } I \text{ to } T_2 \text{ do } C](s, e', a), & \text{falls} \\ \quad \mathcal{B}[I \geq T_2] s e = (\text{false}, e') \\ (s, e', a), & \text{falls } \mathcal{B}[I \geq T_2] s e = (\text{true}, e') \\ \text{Fehler}, & \text{sonst} \end{cases}$$

Bin mir aktuell nicht 100% sicher, weil ich die erste Zuweisung etwas unterschlage. Eigentlich müsste man das $I := T_1$ noch vor der ersten Ausführung vom Befehl C ausführen.

Aufgabe 3

Erweitern Sie die Sprache **WHILE** um den atomaren booleschen Term `eof`. Die informelle Semantik von `eof` lautet: `eof` ist wahr gdw die Eingabe leer ist.

Formalisieren Sie die Semantik von `eof` denotationell.

Idee von Flo:

$\mathcal{B}[\text{eof}](s, e) = (\text{falsch}, e')$, falls $e = b.e'$ mit $b \in \text{BOOL}$ oder $e = E$ oder $e = n.e'$ mit $n \in \text{Zahl}$
 $\mathcal{B}[\text{eof}](s, e) = (\text{wahr}, e')$, sonst

Ich hoffe ich habe alle möglichen Eingabearten abgegriffen.

Aufgabe 4

Programmieren Sie in WHILE (einschließlich eof) einen Algorithmus zur Berechnung der Summe aller Eingabewerte. Beweisen Sie die Korrektheit Ihres Programms anhand der denotationellen Semantik. Diskutieren Sie die Problematik beim Fehlen von eof. **Programm in WHILE:**

```
1 zahl := 0
2 while ¬ eof do zahl := zahl + read
```

Diskussion:

Ohne eof bräuchte das Programm für das Einlesen der kompletten Datei mittels einer Schleife eine Fehlerbehandlung (Exception-Handling), um eine leere Eingabe abzufangen, da das Programm sonst abstürzen würde (read wirft einen Fehler bei leerer eingabe).